

OpenMPI Installation and Setup on Mac

To install OpenMPI on MacOSX, complete the steps below:

1. Press `Command+Space` and type **Terminal** and press **enter (return)** key.
2. Install Homebrew by running in Terminal app:

```
/bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"
```

and press **enter/return** key.

If the screen prompts you to enter a password, enter your Mac's user password (the password when logging in/restarting your Mac) to continue. The password will not be displayed on screen. After typing your password, press ENTER/RETURN key. The command may take several minutes to complete.

It can take several minutes to download and install all the libraries. There will be no loading bar displayed – eventually it will move to installing.

3. Run:

```
brew install open-mpi
```

You can now use `open-mpi`. To confirm OpenMPI has been correctly installed, type the following into Terminal: `mpi_info`

This should print information on your mpi version, similar to the output below.

```
Package: Open MPI brew@Ventura Distribution
Open MPI: 4.1.5
Open MPI repo revision: v4.1.5
Open MPI release date: Feb 23, 2023
Open RTE: 4.1.5
Open RTE repo revision: v4.1.5
Open RTE release date: Feb 23, 2023
OPAL: 4.1.5
OPAL repo revision: v4.1.5
OPAL release date: Feb 23, 2023
MPI API: 3.1.0
Ident string: 4.1.5
Prefix: /usr/local/Cellar/open-mpi/4.1.5
Configured architecture: x86_64-apple-darwin22.4.0
Configure host: Ventura
Configured by: brew
Configured on: Thu Feb 23 04:30:08 UTC 2023
Configure host: Ventura
```

Intel Processor (non M1/M2)

```
Package: Open MPI brew@Ventura-arm64.local Dis
Open MPI: 4.1.5
Open MPI repo revision: v4.1.5
Open MPI release date: Feb 23, 2023
Open RTE: 4.1.5
Open RTE repo revision: v4.1.5
Open RTE release date: Feb 23, 2023
OPAL: 4.1.5
OPAL repo revision: v4.1.5
OPAL release date: Feb 23, 2023
MPI API: 3.1.0
Ident string: 4.1.5
Prefix: /opt/homebrew/Cellar/open-mpi/4.1.5
Configured architecture: aarch64-apple-darwin22.4.0
Configure host: Ventura-arm64.local
Configured by: brew
Configured on: Thu Feb 23 04:30:08 UTC 2023
Configure host: Ventura-arm64.local
```

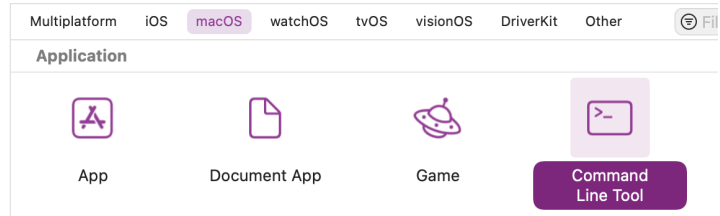
ARM Processor (M1/M2 etc)

*** Pay attention to the 'Prefix' output as this provides the path where MPI is located on your machine, needed for the next step. ***

Option A: Running via IDE using Xcode (recommended)

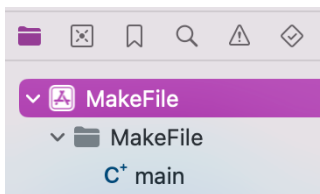
To run C++ using the MPI library on an IDE, first you will need to [download and install Xcode from the Mac Store](#). You will also need to [install command line tools](#). If you already have Xcode installed you can check to [see if the command line tools have already been installed](#).


Once installed, open Xcode and create a new Project > Command Line Tool. Name your project MakeFile, and select C++ as the language.



Next go to File > Project Settings and set Derived Data to 'Project-relative Location'. This makes finding your executable much easier should you wish to locate it in your directory.

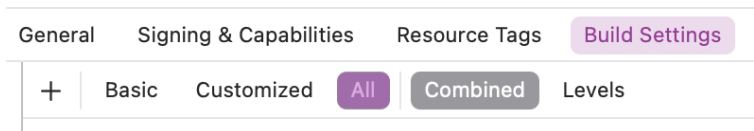
The next step is configuring Xcode to recognise and run the MPI library.



On the top left of the Xcode IDE, click the topmost folder icon  to see the following folder structure.

Click on the name of your project (in this case MakeFile) and open the Build Settings tab on the right.

Ensure that the 'All' option is selected. If 'Basic' is selected you may not see all the submenus.



In 'Search Paths', edit 'Header Search Paths' and 'Library Search Paths'. Use the path given under prefix when you run `omp_i_nfo` in the terminal, and add `/**` to the end of it.

E.g. in my case, the Search paths text is: `/usr/local/Cellar/open-mpi/5.0.3_1/**`

Typically homebrew will install in the Cellar folder by default, **however your path may be different.**

If running on an apple Silicon machine (M1/2/3 etc), your prefix will likely be under the 'opt' folder.

Search Paths	
Setting	MakeFile
Always Search User Paths (Deprecated)	No
Framework Search Paths	
Header Search Paths	<Multiple values>
Debug	
Any Architecture Any SDK	/usr/local/Cellar/open-mpi/5.0.3/**
Release	
Any Architecture Any SDK	/usr/local/Cellar/open-mpi/5.0.3/**
Library Search Paths	<Multiple values>
Debug	
Any Architecture Any SDK	/usr/local/Cellar/open-mpi/5.0.3/**
Release	
Any Architecture Any SDK	/usr/local/Cellar/open-mpi/5.0.3/**

Next you will need to provide flag arguments that point to the custom C++ compiler wrappers MPI provides. This tells Xcode not to use the default compilation command from clang which is 'g++' and to use 'mpic++' (or similar) from MPI instead.

To find your linker flag arguments, open Terminal and type the command:

```
mpic++ --showhelp:compile
```

Add these to Linking > Other Linker Flags. Typically the arguments follow this format:

Intel Processor (Non-M):

```
-I/usr/local/Cellar/open-mpi/*mpi_version*/include  
-L/usr/local/opt/libevent/lib  
-L/usr/local/Cellar/open-mpi/*mpi_version*/lib  
-lmpi
```

ARM Processor (M1/M2/M3 etc):

```
-I/opt/homebrew/Cellar/open-mpi/*mpi_version*/include  
-L/opt/homebrew/opt/libevent/lib  
-L/opt/homebrew/Cellar/open-mpi/*mpi_version*/lib  
-lmpi
```

Linking - General	
Setting	MakeFile
Other Linker Flags	<Multiple values>
Debug	
Any Architecture Any SDK ▾	-I/usr/local/Cellar/open-mpi/5.0.3_1/include -L/usr/local/Cellar/open-mpi/5.0.3_1/lib -lmpi
Release	
Any Architecture Any SDK ▾	-I/usr/local/Cellar/open-mpi/5.0.3_1/include -L/usr/local/Cellar/open-mpi/5.0.3_1/lib -lmpi
Warning Linker Flags	

Xcode will now be able to 'Build' (aka compile) your file when you press ⌘+B, the play button or Product > Run in the menu. Now that Xcode has been configured for compilation, you'll want to be able to run your code as well. You can set up a Scheme in Xcode that allows you to run your code directly in the IDE.

To do so, navigate to Product > Scheme > Manage Schemes and click +. Name your scheme something memorable, for example 'MPI_runner'. Select your new scheme and click Edit.

From the 'Run' tab on the left-hand side go to 'Arguments'. Here you can add command line options and arguments. For example, when running in terminal you would typically use the command: `mpirun -np 4 --oversubscribe /Makefile`

In this case we would need to tell Xcode the three options we want to set as well as the command to be used (mpirun).

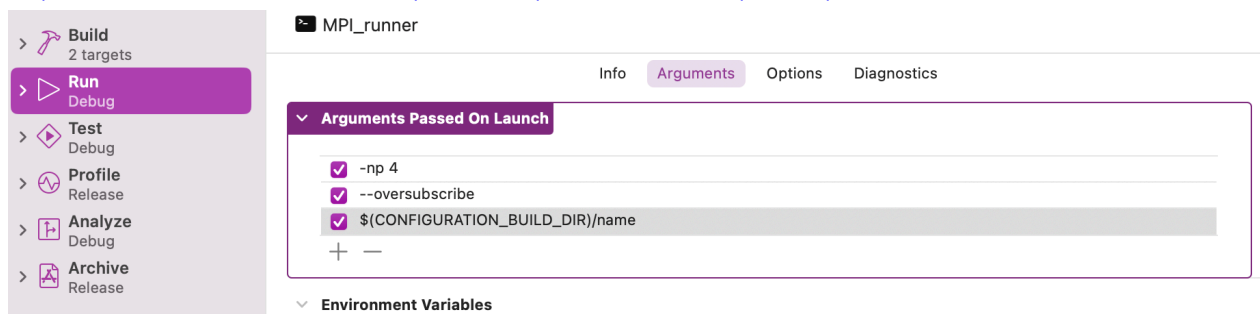
Add these command line options as Arguments Passed on Launch:

-np 4	The number of processes we want MPI to launch. In this case, 4. Change this argument for however many nodes you're wanting to run the program with.
-------	---

<code>--oversubscribe</code>	Indicates that more processes should be assigned to any node in an allocation than that node has slots for. Nodes can be oversubscribed, even on a managed system.
<code>\$(CONFIGURATION_BUILD_DIR)/name</code>	Indicates where the executable file to be run is located. Xcode will look for an executable matching the given name in red (use your project name) in the build folder. The executable is automatically created and saved when you successfully compile your .cpp file.

For more information on command line options for mpirun, visit:

<https://www.ibm.com/docs/en/smpi/10.2?topic=command-mpirun-options>



You can change or add arguments at any time by editing your MPI_runner Scheme. You can also create multiple schemes, for example for running on 1, 4, 8 nodes etc.

Select the 'Info' tab and go to 'Executable'. By default it will have an executable named the same as your Project. In the dropdown, select 'Other' and navigate to your local 'open-mpi/*version*/bin folder. You will need to search in your usr or opt folder, so navigate to your MacintoshHD drive and press `cmd+shift+.` to see your hidden folders. The path will be the same as the 'Prefix' from the previous steps. In the bin folder you should see various executables including mpirun and mpiexec. Select 'mpirun' and close.

Now copy the ['hello world' code](#) from the bottom of this document into your main.cpp class and press play to run. You should see 'Build Succeeded' appear on screen and the following output in the console at the bottom:

```
Hello 0, World: 4
Hello 1, World: 4
Hello 2, World: 4
Hello 3, World: 4
Program ended with exit code: 0
```

* If you see OpenCL errors you may ignore these as they only appear when running via the IDE.

Note: You will need to complete this process for each project you create in order for MPI to work correctly in Xcode. If you prefer not to set up a project in Xcode, you can instead choose to [run from the command line](#).

Option B: Running via the Command Line

Alternatively, you can forgo using Xcode altogether and instead use your chosen IDE to create and save a .cpp file. You will still need [Xcode](#) and the [CL Tools installed](#). You can use any IDE or text editor to build your C++ code, I recommend [Sublime Text](#).

For your first program you can use the [given Hello World script](#). Then using Terminal, navigate to where you have saved the .cpp file and use the following command to compile:

```
mpic++ NameOfFile.cpp -o NameofExecutable
```

This should create an executable file with that name, that you can then run using the mpirun command:

```
mpirun -np NumberOfNodes* --oversubscribe NameofExe*
```

*replace NumberOfNodes with the amount of nodes you'd like to run the program with, and NameOfExe with the executable name you selected at compilation.

Example:

A script named 'hello_world.cpp' is compiled and outputs an executable 'hello_exe' in the same directory using the mpic++ command. The executable is then run on 4 nodes using the mpirun command.

```
jlebron@pegasus CommandLineScripts % ls
hello_world.cpp
jlebron@pegasus CommandLineScripts % mpic++ hello_world.cpp -o hello_exe
jlebron@pegasus CommandLineScripts % ls
hello_exe      hello_world.cpp
jlebron@pegasus CommandLineScripts % mpirun -np 4 --oversubscribe hello_exe
Hello 0, World: 4
Hello 1, World: 4
Hello 3, World: 4
Hello 2, World: 4
jlebron@pegasus CommandLineScripts %
```

mpic++ or mpirun commands not recognised: make sure you have installed open-mpi correctly. You may also want to echo \$PATH to ensure your shell is using the correct compilation wrapper. If the path doesn't include the bin of your open-mpi folder, try this command:

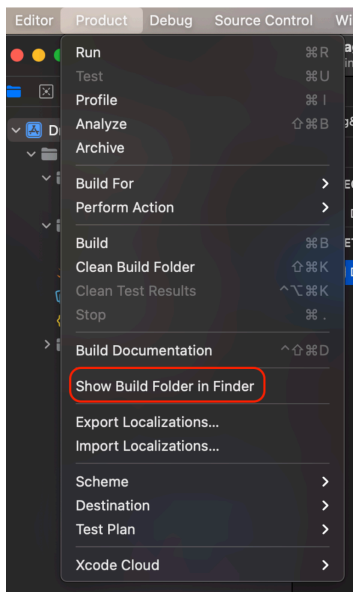
```
PATH=path/to/open-mpi/version/bin:/opt/local/bin:/opt/local/sbin:/usr/local/bin:/usr/bin:/bin:/usr/sbin:/sbin:/opt/local/bin:/opt/local/sbin:/Library/TeX/texbin:/opt/X11/bin:/Library/Apple/usr/bin
```

Replace the red text with your mpi bin, and blue highlight with path as printed from echo \$PATH

Option C: A mix of the above

If you have set up the MPI library in your Xcode project and can Build your c++ code but are unable to run it in the IDE, you may want to choose a mixed approach where you compile using Xcode and run using Terminal.

In this case you will need to navigate to the exact location of your executable file. Depending on your setup, this file might not be easily locatable. If you chose 'Project-relative Location' for your Derived Data, your executable should be in the same directory as your Project Workspace: DerivedData/ProjectName/Build/Products/Debug/



To find your executable, select Products > Show Build Folder in Finder. In Finder, right click on the file to display a dropdown menu. Press the Option key (alt) to display the dropdown option titled 'copy Makefile as Pathname'. This will copy the absolute path of the location of your executable file.

[This link](#) contains more information on copying the pathname.

Once you have the path, open a Terminal window. Use the change directory command 'cd' and the path you just copied to change directory to where your executable file is located. Remember it is an absolute path, not a relative path, so you may have to use cd to navigate to your Users folder first.

To run the file, use the mpirun command, following this template:

```
mpirun -np (number of nodes) --oversubscribe (name of file)
```

For example, to run a file called Makefile using 8 nodes, the command would be:

```
mpirun -np 8 --oversubscribe Makefile
```

Notice the .cpp file is called main.cpp and is stored in a separate folder. The file we are running is not the c++ file, but the executable, typically displayed in Xcode as a black icon.

*[See previous section](#) if mpic++ or mpirun commands are not recognised in Terminal.

Final Step: Hello World

You can now create a HelloWorld MPI program in Xcode. Compile this program, then use the `mpi run` command to run your program with however many nodes you choose.

The correct output would be `Hello World` printed to console the same amount of times as nodes chosen. E.g. if you run with eight nodes, expect `Hello World` to be printed eight times. Your hello world script may also print the world rank (the number of the process currently running) and the world size (the total amount of processes that will run).

Hello World code:

```
#include <iostream>
#include <mpi.h>
int world_size, world_rank;
int main(int argc, char** argv) {
    MPI_Init(NULL, NULL);
    MPI_Comm_size(MPI_COMM_WORLD, &world_size);
    MPI_Comm_rank(MPI_COMM_WORLD, &world_rank);
    std::cout << "Hello " << world_rank << ", World: " << world_size << std::endl;
    // finalise MPI and return control to the OS;
    MPI_Finalize();
    return 0;
}
```